We Claim:

1. In a computer system, a method for implementing and using a filter object which generates an output in response to an input of the filter object, wherein the output of the filter object depends on the input and a state of the filter object, wherein the state of the filter object includes a minimum amount of information necessary to determine the output of the filter object, the method comprising:

providing the filter object, the filter object being represented by equations performed to generate the output in response to the input of the filter object, the equations including the state of the filter object; and

retaining the state of the filter object,

wherein the filter object is implemented and used in a first dynamically typed text-based programming environment.

2. The method of claim 1 wherein the filter object retains a final value of the state obtained as a result of processing the input of the filter object.

3. The method of claim 2 wherein the final value of the state retained in the filter object is used as an initial value of the state for processing the input of the system.

4. The method of claim 1 further comprising the step of resetting the state of the filter object retained in the filter object.

5. The method of claim 1 further comprising the step of presetting the state of the filter object retained in the filter object.

6. The method of claim 1 wherein the output of the filter object is determined depending on a present input and a previous input of the filter object.

7. The method of claim 6 wherein the state of the filter object contains information about the previous input of the filter object.

8. The method of claim 1 wherein the filter object is utilized to generate code to implement a corresponding filter algorithm separate from the filter object implementation.

9. The method of claim 1 wherein the filter object is utilized to generate code to implement a corresponding test bench or filter analysis.

10. The method of claim 8 wherein the generated code can be executed, directly or via a suitable compilation process, on the host machine, but outside the context of the simulation environment in which the filter object executes.

11. The method of claim 10 wherein the generated code is a textual language.

12. The method of claim 10 wherein the generated code is a graphical description language.

13. The method of claim 8 wherein the generated code can be executed, directly or via a suitable compilation process, on the host machine, within the context of the simulation environment on which the filter executes.

14. The method of claim 8 wherein the generated filter code can be executed, directly or via a suitable compilation process, in an environment separate from the computer system used for simulation of the filter object, including an embedded system implementation.

15. The method of claim 14 wherein the generated code is textual language.

16. The method of claim 14 wherein the generated code is a graphical description language.

17. The method of claim 14 wherein the generated code is suitable for use with a software implementation, including use on a general purpose processor, a digital signal processor, or other programmable compute architecture.

18. The method of claim 14 where the generated code is suitable for use with a hardware implementation, such as Field Programmable Gate Array (FPGA), Complex Programmable Logic Device (CPLD), or Application Specific Integrated Circuit (ASIC) device, the language

being VHSIC Hardware Description Language (VHDL), Verilog, or other hardware description language.

19. The method of claim 8 wherein the code is a high-level programming language.

20. The method of claim 8 wherein the code is a low-level machine or assembly language.

21. In a computer-implemented system, a method for generating an output of the system in response to an input of the system, the method comprising the steps of:

specifying a state of the system that includes a minimum amount of information that is necessary to determine the output of the system;

retaining the state of the system in a memory;

providing to the system the state of the system retained in the memory; and

determining the output of the system depending on the input and a state of the system,

wherein the method is implemented in a dynamically typed text-based programming environment.

22. The method of claim 21 further comprising the step of specifying equations that the system performs to generate the output of the system from the input and the state of the system.

23. The method of claim 21 further comprising the step of controlling the state of the system retained in the memory.

24. The method of claim 23 wherein the state of the system retained in the memory is reset to provide a zero initial state to the system.

25. The method of claim 23 wherein the state of the system retained in the memory is set to a particular value entered by a user.

26. The method of claim 21 wherein the state of the system retained in the memory includes a final value of the state obtained as a result of processing the input of the system.

27. The method of claim 21 wherein the state of the system provided to the system includes an initial state of the system for processing the input of the system.

28. A computer readable medium holding instructions executable in a computer that provides a dynamically typed text-based programming environment, wherein the computer generates an output of an object in response to an input of the object, comprising:

providing a class, the object being an instance of the class;

specifying a state of the object that includes a minimum amount of information that is necessary to determine the output of the system, the state being a property of the object; and

determining the output of the object depending on the input and the state of the system.

29. The medium of claim 28 further comprising the step of instantiating the object from the class.

30. The medium of claim 28 wherein the object includes an adaptive filter object.

31. The medium of claim 30 wherein the adaptive filter object includes an adapting algorithm that the adaptive filter performs.

32. The medium of claim 28 wherein the object includes a discrete time filter object.

33. The medium of claim 28 further comprising the step of controlling properties of the object including the state of the object.

34. The medium of claim 33 wherein the state of the object is reset to zero.

35. The medium of claim 28 further comprising the step of inheriting the state of the object from an abstract class.

36. The medium of claim 28 further comprising the step of providing the class with methods which operate on the object of the class.